

# git rebase

Rebase는 커밋을 새로운 부모 커밋으로 옮기는 기능을 제공한다.

**!** Push된 history를 rebase 하지 않도록 주의한다. 이 경우 history를 기반으로 작업하고 있는 개발자들에게 영향을 주게된다.

다음 명령은 현재 working branch의 커밋을 <base> branch로 옮긴다.

```
git rebase <base>
```

git rebase의 기본 동작은 non-interactive이며, interactive mode (--interactive, -i)로 editor를 이용할 수 있다.

```
git rebase -i new-base-branch
```

editor 화면:

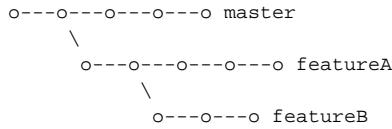
```
pick 2231360 some old commit
pick ee2adc2 Adds new feature
# Rebase 2cf755d..ee2adc2 onto 2cf755d (9 commands)
#
# Commands:
# p, pick = use commit ( )
# r, reword = use commit, but edit the commit message ( )
# e, edit = use commit, but stop for amending ( )
# s, squash = use commit, but meld into previous commit ( )
# f, fixup = like "squash", but discard this commit's log message ( )
# x, exec = run command (the rest of the line) using shell ( )
# d, drop = remove commit ( )
```

editor를 종료하면 선택한 명령이 일괄 수행된다.

--onto 옵션은 브랜치를 rebase 하는 기능을 제공한다.

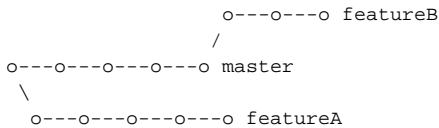
```
git rebase --onto <newbase> <oldbase>
```

featureB branch는 featureA와 의존성이 없어서 master로 rebase하고 싶은 경우,



```
git rebase --onto master featureA featureB
```

rebase 결과:



## Git 가이드

### 1. Git 시작하기

- Git 저장소
  - git init
  - git clone
  - git config
- 변경 저장하기
  - git add
  - git commit
  - git diff
  - git stash
  - .gitignore
- 저장소 점검하기
  - git status
  - git log
  - git tag
  - git blame
- 변경 취소하기
  - git 실행 취소
  - git clean
  - git revert
  - git reset
  - git rm
- Rewriting history
  - git commit -- amend
  - git rebase
  - git reflog

### 2. Git 협업하기

- 동기화하기
  - git remote
  - git fetch
  - git pull
  - git push
- 브랜치 사용하기
  - git branch
  - git checkout
  - git merge
  - 병합 충돌 해결하기 (Merge conflicts)
  - 병합 전략 (Merge strategies)
- Pull request 만들기

